DAO Office Note 1999-004

Office Note Series on
Global Modeling and Data Assimilation

Robert Atlas, Head
Data Assimilation Office
Goddard Space Flight Center
Greenbelt, Maryland

Benchmark and Unit Test Results of the Message-Passing GEOS General Circulation Model

William Sawyer and Alex Wang[†]

Data Assimilation Office, Goddard Laboratory for Atmospheres
Goddard Space Flight Center, Greenbelt, Maryland
[†] *Raytheon Corporation*

Goddard Space Flight Center
Greenbelt, Maryland 20771
August 1999

Abstract

This document contains the benchmarking and unit test results for the message-passing Goddard Earth Observing System General Circulation Model (MP GEOS GCM). The MP GEOS GCM is an atmospheric model which operates on uniform and stretched latitude-longitude grids and is closely related to the shared memory parallel GEOS GCM which is currently in production at the Data Assimilation Office.

The unit tests give an indication that individual components on the MP GEOS GCM are performing as expected and that the results for a given processor configuration are reproducible. The benchmarks reflect the overall performance of the MP GEOS GCM and give an idea of how the application might perform in a production setting. This document also serves as a wrap-up of the parallelization of the GCM for the High Performance Computing and Communication (HPCC) initiative.

Contents

List of Figures

1 Introduction

This document describes the unit test and benchmark results for the message-passing Goddard Earth Observing System General Circulation Model (MP GEOS GCM), version vc7.2. The MP GEOS GCM has been documented extensively in other places [1, 2, 3, 4, 5]. In particular, the design of the MP GEOS GCM can be found in [6]. The MP GEOS GCM is, in brief, an advanced message-passing atmospheric model, supporting uniform and stretched latitude-longitude grids, utilizing such Fortran 90 features as modules, function overloading, derived types and dynamic allocation and using the Message Passing Interface (MPI) for communication. It adheres to a Single-Program Multiple-Data (SPMD) paradigm: the MP GEOS GCM splits up the lat-lon view of the world into local rectangular regions, each assigned to a processor. That means that the GCM gridded data are generally distributed with a column decomposition (for 3D arrays) or a "checkerboard" decomposition for 2-D arrays (see Figure 1).

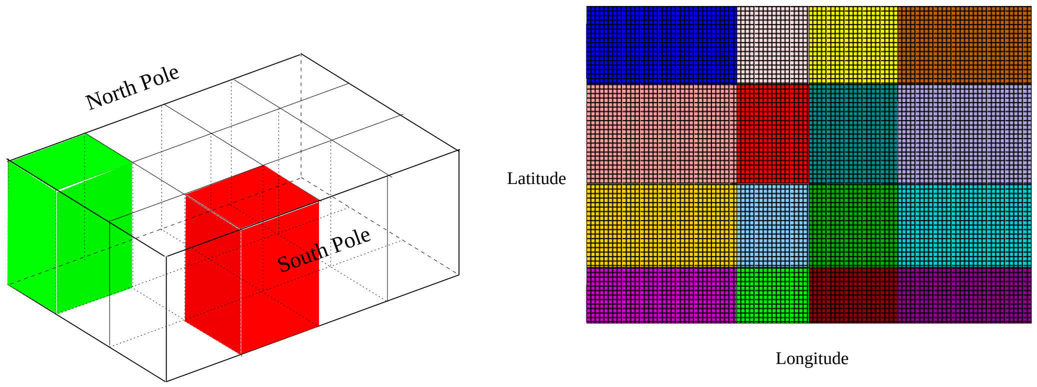

Figure 1: Three-dimensional arrays are generally broken up into columns, each assigned to a processor. Similarly two-dimensional arrays, are distributed as local rectangles. Note that the latitude and longitude "cuts" need not be uniform, and the size of the column or rectangle can vary from processor to processor.

The MP GEOS GCM unit tests are generally comparisons between the message-passing versions of individual code components and their sequential counterparts. These tests indicate the magnitude of the differences between these versions, and verify that message-passing code run a given processor configuration gives reproducible results. Routines unique to the MP GEOS GCM have their own dedicated tests. Some of the unit tests can also serve as benchmarks of performance-critical software components. The unit tests were performed on an SGI Origin 2000 and the Cray T3E. The unit test results and some component benchmarks are given in Section 2.

The GCM benchmark is a 1h simulation of the GCM (without history output and data analysis) for $144 \times 91 \times 48$ (which corresponds to $2^o \times 2.5^o \times 48$ if a uniform grid is employed) and $360 \times 181 \times 48$ ($360 \times 181 \times 48$ for a uniform grid) resolutions. The benchmarking was performed on four different modern supercomputers: the SGI Origin, Cray T3E, NEC SX-4, and IBM SP3. These are briefly described in Section 3 before the presentation of the overall GCM benchmarks.

A list of the performance enhancements performed on the MP GEOS GCM in its 2+ year history are given in Section 4. A discussion of all the results can be found in Section 5 as well as some comments about the current state of the code and its potential for attaining more ambitious performance goals.

2 Unit Tests and Component Benchmarks

The unit tests are meant to ensure that individual parallel components perform as expected. Since sequential unit tests were not available to adapt to the parallel code, these unit tests generally run the parallel code and sequential code (from the official GCM releases) simultaneously and compare the results to a given tolerance. In [7] it is argued that zero differences cannot be expected in every case. On the other hand, in the worst case only round-off differences are expected. Furthermore, reproducibility for a given processor configuration (a requirement for MP GEOS GCM) can be achieved thanks to the deterministic execution of the parallel code.

In addition to their role in quality assurance, the unit tests were also designed to provide component benchmarks, some of which have been presented already [2, 8, 9]. For such benchmarks, the sequential comparison is turned off. Such component benchmarking made it possible to isolate potential performance bottlenecks early on and concentrate on their optimization. Such critical components are mentioned in the subsequent sections along with the unit test results.

The unit tests are split up into groups for each of the GCM software libraries which they validate: the *Hermes*[1] grid transformations, the dynamical core, the utilities, the atmospheric science software, and the earth- and the ocean-related software. The unit tests were performed on an SGI Origin at $144 \times 91 \times 48$ resolution unless indicated otherwise. The tests were run on the same processor configurations several times in order to determine whether the output is **reproducible**, which it should be in all cases due to the deterministic design of the parallel version. The unit tests were also tested on various processor configurations to find the maximum relative difference or **max. rel. diff.** from the sequential version.

2.1 Hermes Unit Tests

The Hermes library contains a number of routines to transform data between the various grids used in the GCM. Since these routines operate on distributed data with the column decomposition and since there are numerous horizontal dependencies in the transformations, communication is unavoidable.

Name	Routine tested	Reproducible?	Max. Rel. Diff.
HermesUnitTest	C-to-A transform	Yes	1.0E-15
HermesUnitTest	A-to-C transform	Yes	1.0E-15
HermesUnitTest	rotate forward	Yes	5.0E-15
HermesUnitTest	rotate backward	Yes	5.0E-15
HermesUnitTest	comp. to geo.	Yes	1.0E-13
HermesUnitTest	geo. to comp.	Yes	1.0E-13
HermesUnitTest	polar wind	Yes	1.0E-15
HermesUnitTest	wind transform	Yes	1.0E-15

The C-to-A and A-to-C transformations utilize bi-cubic interpolation to determine A-grid values from a C-grid and vice versa. The pole rotation algorithm is special in that it is not a direct parallelization of the sequential routine. Since the operation applies a linear transformation on one field to obtain the values on another, the pole rotation was split up into its two constituent parts: the definition of the rotation coefficients and their application on a vector. The latter operation was deemed one of several kernel operations [8] which might be used in other sections of the code. The

[1] *Hermes* can be thought of as the "messenger" between different grids; credit for the name goes to Arlindo da Silva.

computational to geophysical and geophysical to computational grid transformations stretch and/or rotate all the significant fields from one frame to the other and make repeated calls to the rotation routines.

Component Benchmarks There was some indication from the beginning of the MP GEOS GCM project that the rotation would pose a bottleneck to parallel performance. Extensive prototype benchmarking indicated that indeed the rotation was highly latency-sensitive and its performance could be enhanced considerably by the use of low-latency Cray-proprietary SHMEM message-passing for the critical communication (see Section 4 for further discussion). Figure 2 indicates the improvement achievable with SHMEM on the Cray T3E.

With the column decomposition used for 3-D fields, the C-to-A and A-to-C tranformations are textbook examples for boundary exchange between neighboring processors. Thanks to overlapping of communication with the local computation, excellent scalability can be obtained (see Figure 7).

Prototypes for the parallel versions were constructed in 1997 and were presented at the Scientific Advisory Board meeting [9], May, 1998. Version vc6.6 encompassed a complete rewrite of the Hermes library to support stretched grids, and the parallel versions were re-implemented in Spring 1998.

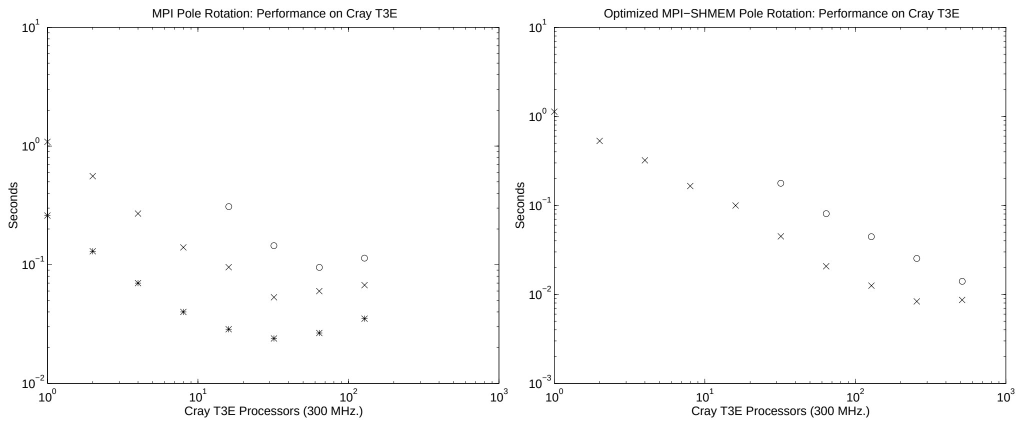

Figure 2: The pole rotation tends to create an irregular and imbalanced communication pattern with relatively small messages. High MPI latencies on the Cray T3E affect performance adversely at higher numbers of processors (left). Although it is not portable, the use of a low-latency, SGI/Cray-proprietary, one-way SHMEM communication can ameliorate the situation (right).

2.2 Utilities Unit Tests

The message-passing utilities[2] are unique to the MP GEOS GCM. These utilities consist of the PILGRIM facilities described extensively in [8]. There are facilities to pack and unpack message buffers, to create, define and destroy data decompositions, to perform high-level communication, to redistribute data between different data decompositions, and to perform sparse linear algebra. Each of these is assigned an independent Fortran 90 module, and for each of these a unit test is available.

[2] To be distinguished from the GCM and land-surface model utilities from the sequential version which were adopted largely unchanged in the MP GEOS GCM.

Name	Module tested	Reproducible?	Max. Rel. Diff.
BufferTest	Buffer pack/unpack	Yes	N/A
DecompTest	Decompositions	Yes	N/A
ParUtilitiesTest	Parallel utilities	Yes	N/A
RedistributeTest	Redistribution	Yes	N/A
SparseTest	Sparse linear algebra	Yes	1.0E-15

Since there are no sequential counterparts to these components, the unit tests generally check for consistency in the data after several manipulations. For example, a data field needs to remain the same after being scattered to and gathered from all the processors (ParUtilitiesTest) or redistributed from one decomposition to another and then back again (RedistributeTest).

Component Benchmarks In order to serve as a component benchmark, the sparse utilities test, along with several consistency checks, performs the application of a pole rotation to 2-D and 3-D fields. It compares the result with the sequential GCM `rotate_f` and `rotate_b` routines, which define the forward or backward transformation (during the first execution of the routine only), and then apply it to a given field. In order to act as a benchmark the sequential-parallel comparison can be turned off, and the code run in an optimized framework. Component benchmarks on the Cray T3E in Figure 3 indicate a successful parallelization, although low-latency Cray-proprietary SHMEM one-way communication primitives were used for optimization.

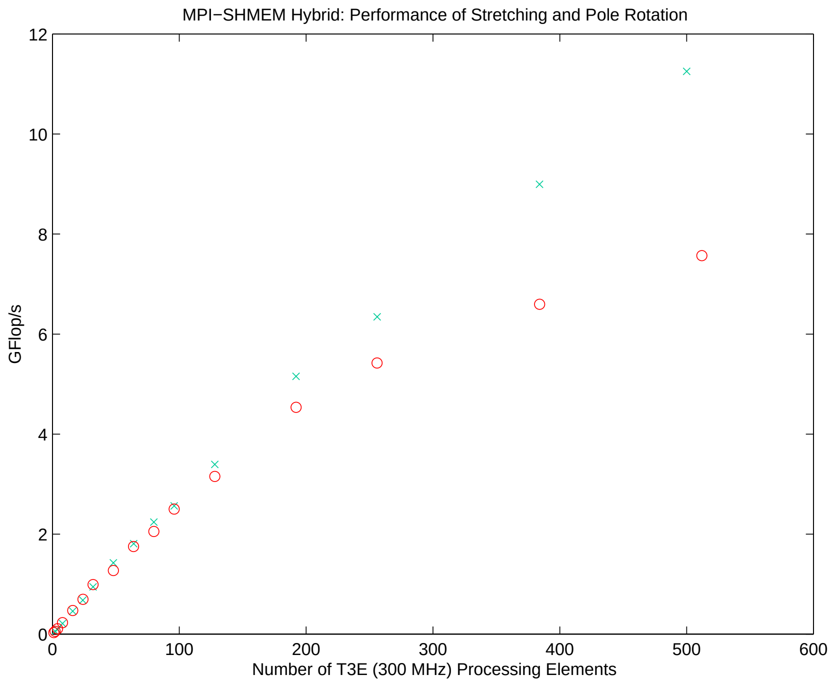

Figure 3: The sparse utilities test also serves as a benchmark for the pole rotation algorithm. After the interpolation coefficients are defined they are applied simultaneous to all levels of a three-dimensional field with $144 \times 91 \times 48$ (**o**) and $360 \times 181 \times 48$ (**x**) resolution. Performance of this optimized version scales to nearly the full extent of the Cray T3E.

2.3 Dynamics Unit Tests

The dynamics posed much of the difficulty in the GEOS GCM parallelization. Its components, such as the dynamical core [10], the high-latitude filter [11] and the Shapiro filter require complicated communication to function equivalently to the sequential code.[3] The stretched grid high-latitude filter AVRXG requires extensive communication along latitudes. The dynamical core requires repeated exchange of boundary regions between processors ("ghosting"). The Shapiro filter requires several layers (depending on the filter order) of boundary regions to be exchanged at once.

The unit tests all compare parallel version results with the sequential version. Due to the local nature of the the Shapiro filter it is possible to maintain the order of arithmetic operations needed to ensure zero differences. Unfortunately this is not possible in the high-latitude filter or in the dynamical core, and round-off differences accumulate.

Name	Routine tested	Reproducible?	Max. Rel. Diff.
AVRXGUnitTest	high-latitude filter	Yes	3.0E-13
DycoreUnitTest	Dynamical Core	Yes	6.0E-12
ShapiroUnitTest	Shapiro Filter	Yes	0.0

Component Benchmarks The performance of a message-passing dynamical core (Figure 4), a preliminary version of ongoing work at Goddard by Suarez and Schaffer, saturated at 64 processors for 144×91 and at 128 processors for 360×181 resolution. This version operated on one vertical level at a time and thus ghosted only 1-dimensional boundary regions, creating many very short messages.

The performance, even with Cray-proprietary SHMEM message passing was too limited for the DAO's requirements. The GCM vc6.5 dynamical core was therefore re-parallelized entirely in Summer 1998 to improve performance and to keep pace with DAO scientific development.

The resulting performance in Figure 5 was considerably better, although slightly deceptive: the high processor executions perform considerably more floating point operations than the low processor runs, due to the computation involved in maintaining consistency in the boundary regions. This effect can be seen in the absolute execution times for the $360 \times 181 \times 43$ resolution in Figure 6 which stagnate slightly at higher numbers of processors.

The same stagnation effect is even more extreme in the Shapiro filter. Boundary regions up to 4 layers deep need to be maintained in the Shapiro filter, meaning that for large numbers of processors, a very large portion of the calculation could be involved in maintaining these regions. Figure 7 illustrates that the Shapiro filter execution time stagnates at 32-128 Cray T3E processors, depending on the resolution. Although the Shapiro filter does not account for a large percentage of the execution time at low numbers of processors, it could become a considerable portion at high processor numbers.

The stretched grid high-latitude filter (AVRXG) requires that entire latitudes be consolidated on all processors containing the latitude. Although there are several approaches, this entails a collective communication on a "row" of processors. Figure 8 indicates good scalability in the absolute execution time, however again this is deceptive. The best processor configurations contain not more than eight processors longitudinally, a limitation when the code is ported to very high numbers of processors.

[3] Although not officially part of the dynamics, the pole rotation / stretching algorithm is closely related and is also one of the most complex parallel components.

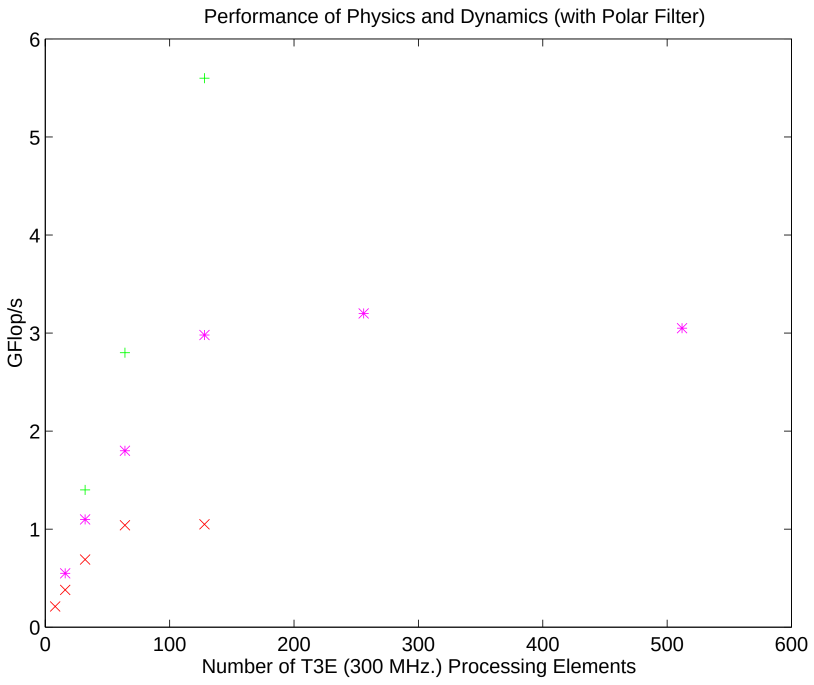

Figure 4: The graph depicts the GFlop/s performance of the vc5.9 dynamical core. This code operates on one vertical level at a time and requires many small messages. Cray-proprietary SHMEM primitives were used to reduce latency, but still the code scales only to about 64 processors for $144 \times 91 \times 70$ (**x**) and to about 128 processors for $360 \times 181 \times 43$ (*****) resolutions. For comparison, the $144 \times 91 \times 70$ ensemble physics (**+**) is given for up to 128 processors.

2.4 Atmospheric Model Unit Tests

The atmospheric model software which goes into the `libatmos.a` library makes up most of the GCM. In particular, in contains all the software for the atmospheric model (except for the dynamics, which has been moved elsewhere for easier software management) including the physics, the drivers for both the physics and dynamics, and the time stepping. It also contains routines to read and write the restart file and rotate and/or stretch the data to the appropriate grid.

The chemistry unit test tests the chemistry model and the related interpolation routines for water vapor and ozone. Zero differences were expected and achieved.

Since the physics (short- and long-wave radiation, turbulence, moist processes, and gravity wave drag) has only vertical data dependencies, zero differences were anticipated in all cases. The unit tests compare the same (sequential) code run in two different ways, first with one processor operating on the global data, and then with multiple processors each working on a local “column” of data. The multiple data are gathered and compared with the sequential data.

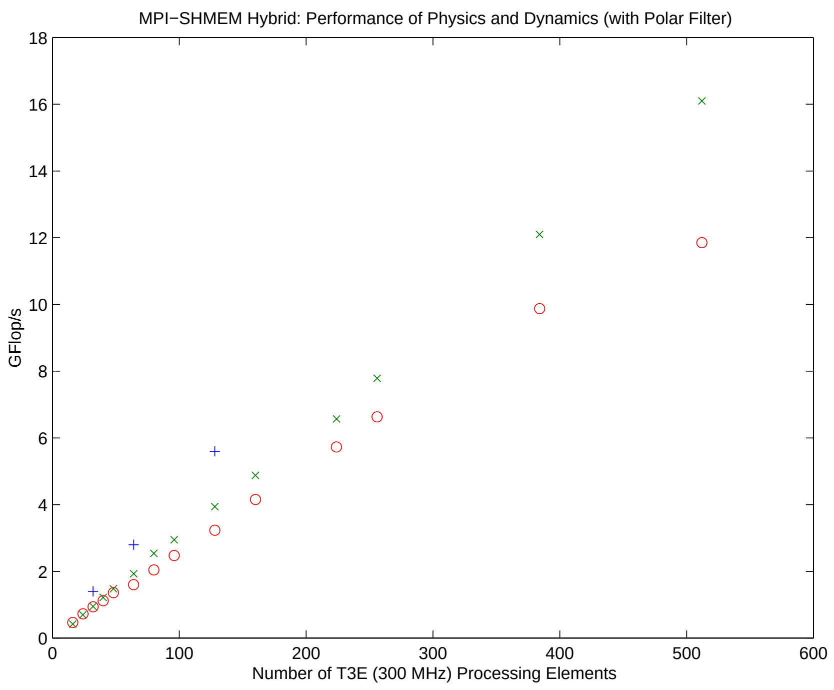

Figure 5: The graph depicts the GFlop/s performance of the dynamical core re-parallelized in Summer 1998 based on GCM vc6.5. This code exchanges boundary information over all levels simultaneously, creating far fewer, but longer, messages than vc5.9. Cray-proprietary SHMEM primitives were used to reduce latency, and the code scales well for $144 \times 91 \times 70$ (**o**) and $360 \times 181 \times 43$ (**x**) resolutions. For comparison, the $144 \times 91 \times 70$ ensemble physics (**+**) is given for up to 128 processors. The good GFlop/s performance is deceiving, however, as the high processor executions perform considerably more floating point operations than the low processor runs, due to the computation involved in maintaining consistency in the boundary region.

Name	Routine tested	Reproducible?	Max. Rel. Diff.
ChemUnitTest	Chemistry model	Yes	0.0
GWDUnitTest	Gravity wave drag	Yes	0.0
LWUnitTest	Long-wave (LWRIO)	Yes	0.0
MoistUnitTest	Moisture (MOISTIO)	Yes	0.0
RestartRotateTest	Restart rotation	Yes	6.0E-16
RestartUnitTest	Restart read/write	Yes	0.0
SWUnitTest	Short-wave (SWRIO)	Yes	0.0
TurbUnitTest	Turbulence (TURBIO)	Yes	0.0 (see text)

The only surprise in the results was the turbulence. In general the sequential and parallel versions give non-negligible differences. After some investigation it was determined that the turbulence calculates thresholds to determine when it can cut-off vertical calculation (in order to minimize floating point operations). These thresholds are different on one processor with the global data set, or a processor with a local subset. These thresholds are in fact also a function of the "strip size" — the number of vertical profiles used in the calculation. At the suggestion of A. Molod, the sequential/parallel comparison was made with strip size 1. This configuration, although not necessarily efficient, gives zero differences. Later benchmarking runs used a strip size optimized for best cache performance.

The restart unit test reads and distributes the restart file and then gathers and writes it out to a file. Naturally the bitwise identical file is expected for both sequential and parallel versions. The restart

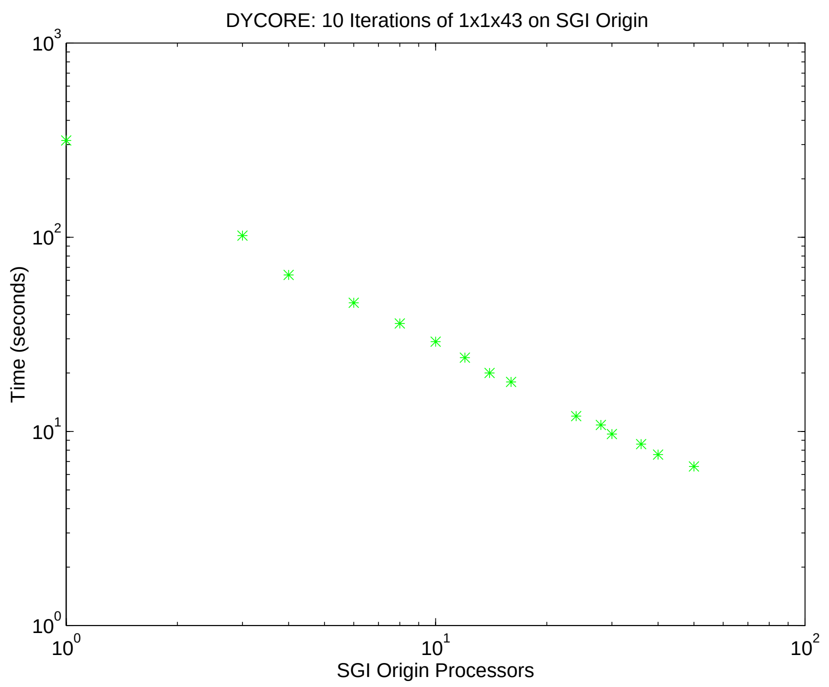

Figure 6: The graph at left depicts the execution times of ten iterations of the dynamic core at $360 \times 181 \times 43$ ($\mathbf{x}$) resolution on the SGI Origin 2000. The MPI code also utilizes SHMEM for the time-critical exchange of boundary regions. The scaling tails off slightly near to the full machine size. One reason for this is the increase in overall floating point operations which are needed to maintain the consistency of the boundary regions.

rotate reads in the restart, distributes it, rotates and/or stretches it according to given parameters, then gathers and writes it to a file. As mentioned in previous sections, the rotation algorithm gives round-off differences, and thus the RestartRotateTest does also.

Component Benchmarks The physics is of particular interest, since it takes up a non-negligible portion of the overall computation. Tests on the SGI Origin 2000 (see Figure 9) indicate the distributed memory versions of the long- and short-wave radiation code scale well to large numbers of processors for both the $144 \times 91 \times 48$ and $360 \times 181 \times 48$ resolutions. The performance of the distributed memory versions compare favorably to the shared memory parallel (multitasked) versions. The gravity wave drag and moist processes take 1–2 orders of magnitude less time on one processor, but they scale much more poorly, and may present a problem at higher numbers of processors.

2.5 Earth/Ocean Unit Tests

The ocean and earth models are currently simple routines which read and distribute boundary condition data. The only real complexity involved is in the distribution of the tile-space arrays in the earth model, whose decomposition is quite irregular [3]. The ocean model unit test compares the parallel ocean model with the sequential. The earth model unit test naturally tests the earth model, but it also has to call the ocean model due to the way the earth code is structured. In both unit tests, zero differences were anticipated and achieved. Since the ocean and earth models are called once or only a few times per run, component benchmarks were not performed.

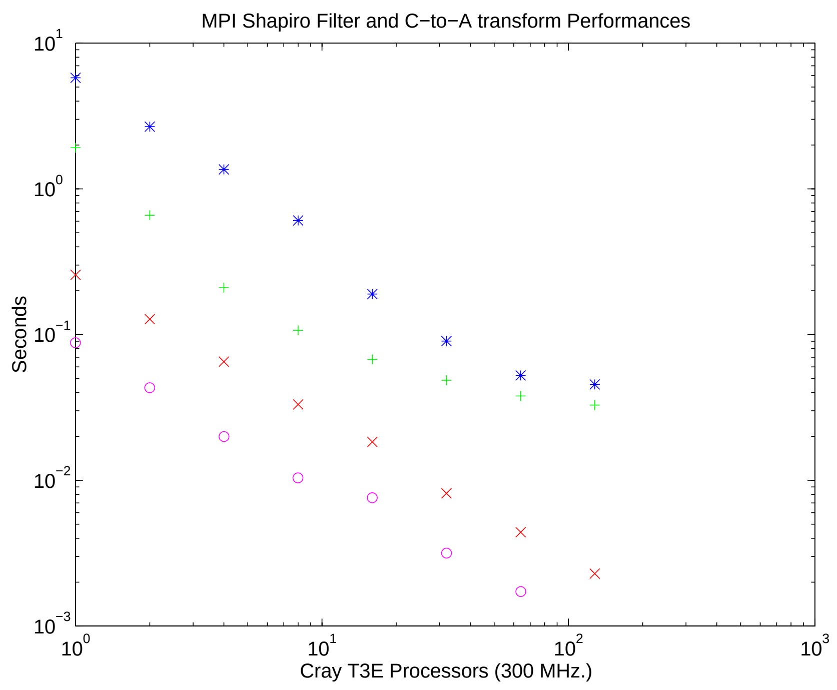

Figure 7: The Shapiro filter filter requires that up to four boundary layers (column "faces") be allocated and maintained during the calculation. At additional calculation becomes noticable at high processor numbers for both $144 \times 91 \times 70$ (+) and $360 \times 181 \times 43$ (*) resolutions.

Name	Routine tested	Reproducible?	Max. Rel. Diff
EarthUnitTest	Earth model	Yes	0.0
OceanUnitTest	Ocean model	Yes	0.0

3 Benchmarks

Of the many possible benchmarking issues we consider those which are most relevant to the DAO's operations. The primary interest is in throughput, in particular the number of days of simulation which can be performed per day of wall clock time. The scalability to large numbers of processors is also a consideration to the DAO as it may help with planning for future platform configurations in the transition to higher resolutions. Finally, the number of floating point operations per second (generally in billions, i.e. GFlop/s) is of tangential interest in understanding the utilization of the theoretical peak machine performance.

There are certain limitations to the MP GEOS GCM code. It is based on version vc7.2 of the GEOS GCM, but it does lack certain components which are non-trivial in terms of execution time. In particular, the parallel output concept (GPIOS) or "history" has not yet been integrated. While the parallel history has been prototyped and the initial benchmarks have been encouraging [12, 13, 14], it was not yet possible to integrate that work. In addition, the Lin-Rood advection scheme was not implemented,[4] and neither was the tracer-fill scheme for user-defined tracers. On the other hand, some elements of vc7.3 are already present in the MP GEOS GCM in particular the new stretched grid high-latitude filter of Takacs [11, 15].

[4]Lin-Rood advection is an integral part of the Finite-Volume Community Climate Model (FVCCM) now being parallelized in an independent project.

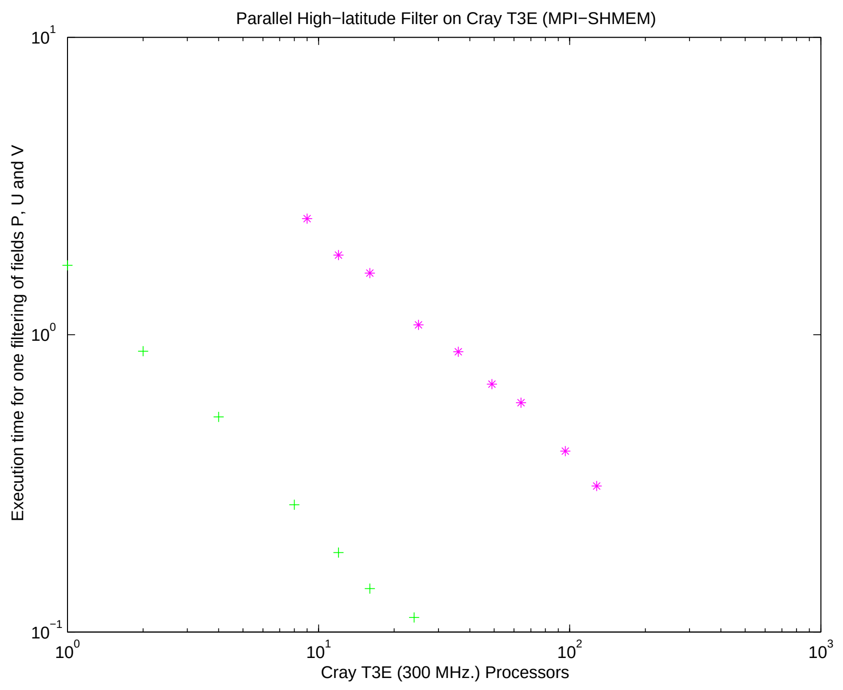

Figure 8: The stretched grid high-latitude filter is applied to entire latitudes near the pole regions. This entails collecting individual latitudes on a processor. The execution time scales well for for both $144 \times 91 \times 70$ (+) and $360 \times 181 \times 43$ (*) resolutions, but this assumes that the data decomposition is finer in latitude than longitude, i.e., that the problem is not split into more than eight processors longitudinally. This could be a limitation for high processor runs.

The simulation employs the leap-frog time stepping scheme, not the Matsuno scheme which has been used in data assimilation mode and which requires somewhat more computation. Thanks to algorithmic adjustments by Takacs [16], the leap-frog time stepping scheme has now supplanted the Matsuno scheme for data assimilation also.

The GCM gridded data are generally distributed with a column decomposition (for 3D arrays) and or a "checkerboard" decomposition for 2-D arrays (see Figure 1). The "tile-space" for the land-surface model is distributed in an irregular fashion which can be derived from the checkerboard decomposition [3]. There is freedom to make different number of "cuts" in latitude and longitude resulting in two-dimensional processor configurations.[5] For example, if 7 processors are allocated longitudinally and 9 latitudinally, the result is a 63 processor configuration on which the problem is calculated. The configuration can be adjusted to find the optimal configuration for a given number of processors.

There is further flexibility in that the "cuts" in latitude and longitude do not need to be regular, that is, the rectangle local to a processor can be of variable size. Therefore the local problem size can be varied to partially compensate for load imbalance inherent in the problem. The only limitation is that each processor contains at least 4 points in each direction (a requirement of the Shapiro filter and the pole calculations). Since the load imbalance is a complex function of many different factors, this feature has not yet been exploited, and the current algorithm balances the number of points on each processor to the largest extent possible.

In all cases, the full GCM model is run with pole rotation but without history. The expense of

[5]This processor grid should be considered virtual since it usually does not coincide with the physical placement or topology of the processors on the underlying platform.

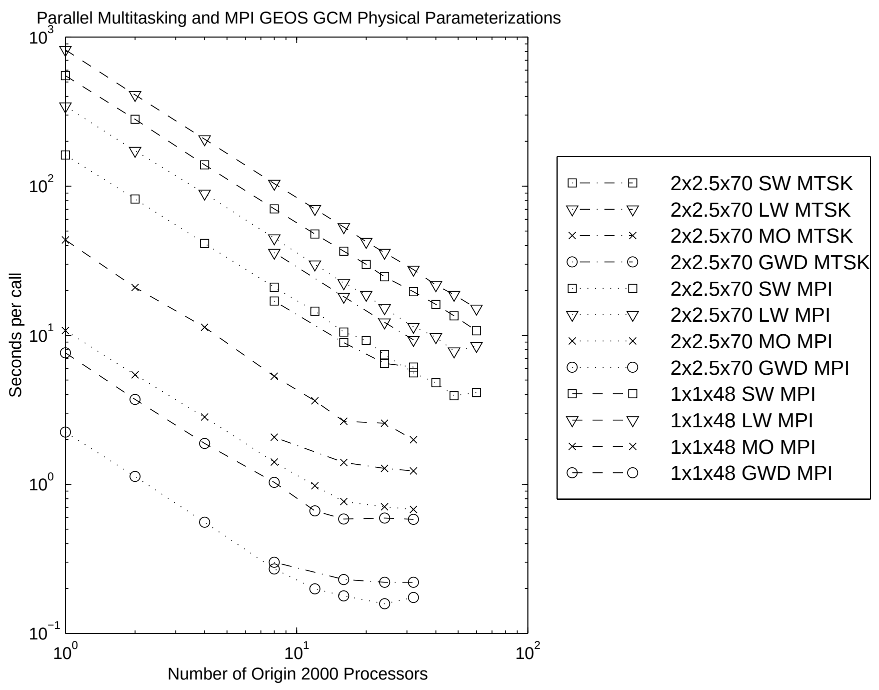

Figure 9: The GCM physical parameterizations have only vertical dependencies and therefore require no communication. The timings indicate that the computationally intensive long-wave (LW) and short-wave (SW) radiation scale well in the number of processors for 360 × 181 and 144 × 91 resolution. The moist processes (MO) and gravity wave drag (GWD) take only a fraction of the execution time, but scale much more poorly. The shared memory parallel or "multitasked" (MTSK) versions, benchmarked on a slightly faster machine, scale considerably less well than their domain decomposed (MPI) counterparts.

the pole rotation is included in the dynamics iteration cost. No stretching is performed on the computational (dynamics) grid. All calculations are with 8-byte reals.

3.1 SGI Origin 2000

The SGI Origin 2000 cluster at NASA Ames Numerical Aerodynamic Simulation group consists of several 32 node (64 processor) machines. Each node contains two MIPS RISC R10000 64-bit CPU running at 195 or 250 MHz (the later system was used for benchmarking). Each processor has 512 Mbytes of memory, a 32 Kbyte primary cache and a 4 Mbyte secondary cache.

Resolution: $144 \times 91 \times 48$ Time-step: 3 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
1	31.94	241.29	7.85	608.97	0.05
2	35.30	122.68	4.93	330.84	0.10
4	26.25	60.52	2.40	155.95	0.20
9	20.69	28.22	1.09	71.16	0.45
15	19.55	18.60	0.80	48.46	0.67
20	19.54	14.50	0.64	38.85	0.85
24	19.16	12.93	0.57	34.21	0.96
28	18.96	10.52	0.49	28.82	1.14
36	18.94	9.45	0.49	26.42	1.26
45	19.73	8.50	0.47	24.25	1.39
49	20.05	7.15	0.47	21.98	1.54
63	21.73	7.52	0.48	22.52	1.52

Resolution: $360 \times 181 \times 48$ Time-step: 2 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
1	175.16	1230.49	46.12	4196.89	0.04
2	222.07	620.28	27.90	2269.54	0.08
4	162.31	313.02	12.72	1059.18	0.17
9	90.52	151.40	7.99	567.28	0.32
15	87.36	87.36	3.77	294.26	0.63
20	62.35	68.10	2.83	227.67	0.81
24	57.38	57.23	2.43	191.33	0.98
28	54.72	49.52	2.06	171.56	1.13
36	48.77	41.71	1.65	133.64	1.41
45	48.48	32.86	1.43	111.77	1.68
49	54.20	30.45	1.36	102.80	1.84
63	50.37	26.60	1.23	92.71	2.04

We have also had limited access to an 32 processor R12K (300 MHz) Origin 2000 at NASA Goddard.

Resolution	$144 \times 91 \times 48$			$360 \times 181 \times 48$		
Compute PEs	Init. (sec.)	1h (sec.)	GFlop/s	Init. (sec.)	1h (sec.)	GFlop/s
4	26.94	134.76	0.23	N/A	N/A	N/A
9	22.45	61.51	0.52	N/A	N/A	N/A
15	21.54	40.08	0.82	85.57	281.40	0.66
25	22.05	27.78	1.18	81.69	166.41	1.12
30	22.91	24.76	1.33	81.41	139.41	1.34

3.2 Cray T3E

The HPCC/ESS CAN Testbed at NASA Goddard, an SGI/Cray T3E, has 1024 300 MHz DEC Alpha processors, 128 Mbytes (16 Mwords) of memory per processor. The processor has a theoretical peak performance of 600 MFlop/s per processor, or one addition and one multiplication simultaneously per clock cycle.

The small local memory on this distributed memory machine makes it difficult to run high resolution cases. There was not adequate memory to run cases on processor configurations smaller than those given in the following tables.

Resolution: 144 × 91 × 48 Time-step: 3 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
20	34.47	38.30	0.99	89.98	0.60
24	26.23	33.77	0.88	77.42	0.69
30	27.14	27.32	0.75	67.08	0.84
36	26.34	23.25	0.66	56.40	1.01
49	26.29	15.75	0.54	41.13	1.31
63	30.32	15.17	0.48	37.15	1.46
81	29.26	11.89	0.43	31.10	2.00
99	29.52	11.49	0.42	28.88	2.07
126	30.73	8.07	0.41	23.30	2.77
144	34.96	8.03	0.41	22.93	3.10

Resolution: 360 × 181 × 48 Time-step: 2 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
168	220.84	23.90	0.97	82.78	3.87
195	175.11	23.19	0.96	77.24	4.03
224	128.90	20.07	0.94	72.07	4.64
255	127.84	16.51	0.91	65.79	4.96
288	180.39	16.36	0.94	63.83	5.51
360	145.98	12.92	0.99	61.14	5.60
400	174.48	12.90	1.05	60.63	6.01

3.3 IBM SP

The IBM Scalable Power Parallel (SP*) installed temporarily at Goddard is a distributed memory MIMD parallel computer consisting of 16 RS/6000 processor nodes, which communicate via a multi-stage interconnect (the SP Switch). Each node contains two Power3 (200 MHz) processors in an SMP configuration. The nodes have a XX MB cache. These nodes communicate with each other by sending and receiving packets through the SP Switch, which provides a bi-directional data-transfer rate of 480 MB/second between Power3 node pairs.

The IBM SP only has shared memory on a node with two processors. Over all the nodes it is a distributed memory machine, and thus the test cases do not "fit" into all processor configuration. Like the Cray T3E There was not adequate memory to run cases on processor configurations smaller than those given in the following tables.

Resolution: 144 × 91 × 48 Time-step: 3 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
4	26.04	70.71	2.44	162.85	0.19
9	58.44	34.12	1.30	81.32	0.39
16	23.84	20.04	0.89	52.85	0.62
25	61.63	13.49	0.67	38.37	0.86
30	63.05	11.71	0.60	34.81	0.95

Resolution: $360 \times 181 \times 48$ Time-step: 3 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
15	136.67	98.67	4.08	309.37	0.60
25	139.63	61.17	2.81	210.19	0.90
30	126.79	53.29	2.48	184.64	1.01

3.4 NEC SX-4

The SX-4 at the Atmospheric Environment Service (AES) of Environment Canada in Dorval, Quebec was made available to us through a collaboration with Recherche en Prévision Numérique (RPN). Although the system contains three nodes with a total of about 80 processors, we limited our activities to a development node with 14 processors. The SX-4 is an air-cooled shared memory machine. Clock speed of the SX-4 is 8ns (125MHz). Each processor of the SX-4 consists of a scalar unit and a vector unit. The scalar unit is a superscalar architecture. There are 128 64-bit scalar registers per CPU. The vector unit of each processor consists of 8 parallel vector sets of 4 pipes, 1 add/shift, 1 multiply, 1 divide, and 1 logical. A single processor thus can attain a peak vector performance of 2 GFlop/s. Main memory configuration of the CSCS SX-4 consists of 8 GBytes of Synchronous Static Random Access Memory (SSRAM). This memory has a 15ns cycle time.

The SX-4 has without question the fastest processor of all the benchmark platforms. Utilizing it depends entirely on the successful vectorization of the code, a task which is not necessarily straightforward. Initial runs, in fact, yielded performances only a factor 2 faster than the SGI. Optimizations increased the performance almost three-fold.

Resolution: $144 \times 91 \times 48$ Time-step: 3 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
1	47.13	62.07	0.82	119.40	0.26
2	45.16	32.51	0.50	66.86	0.47
4	41.92	16.47	0.42	37.79	0.84
6	40.28	11.27	0.35	26.82	1.19
8	40.23	8.69	0.29	21.90	1.46
10	39.75	7.19	0.28	18.80	1.71
12	41.71	6.27	0.25	17.01	1.90

Resolution: $360 \times 181 \times 48$ Time-step: 2 min.					
Compute PEs	Init. (sec.)	First (sec.)	Dyn. (sec.)	1h (sec.)	GFlop/s
1	193.27	300.06	3.06	645.11	0.28
2	220.98	157.24	1.65	345.61	0.53
4	171.38	79.36	0.96	182.43	1.02
6	181.62	55.32	0.76	127.63	1.45
10	170.59	32.61	0.52	81.81	2.27
12	163.49	27.16	0.46	70.26	2.65

4 Optimizations

The code has come a long way since the first prototype version (based on vc5.9) completed in Nov. 1997. The first benchmarks (see Figure 4) indicated that while the physics scaled acceptably (in spite of low overall performance), the dynamical core performance saturated around 64 processors for 144x91x70 resolution and at 128 processors for 360x181x43 resolution. Since then many enhancements have been integrated into the code.

- Various compile flags were tried and the best chosen for the benchmarking.
- The parallel vc5.9 dynamical core scaled poorly. It was determined that the one-level-at-a-time employed in this core was creating many short messages and latency effects were saturating performance on 64 Cray T3E processors. In Summer 1998, the dynamical core from vc6.5 was completely reparallelized, and communication was aggregated over all levels. The GFlop/s performance in Figure 5 reflects this optimization.
- Even with the large messages employed through the previous enhancement, latencies were still an issue in the code, particularly in the pole rotation algorithm. For better performance on the Cray T3E, Cray-proprietary SHMEM code was added to the code for the critical communications. An MPI/SHMEM hybrid can be chosen at compile time by specifying a preprocessing flag, otherwise the code is compiled MPI-only.
- The major rewrite to the sequential Hermes transformations in vc6.4 necessitated a corresponding rewrite of the parallel versions of these routines. During this rewrite the C-to-A and A-to-C transformations were implemented with overlapping of communication and calculation.
- Jim Taft has written an interface to the SGI fast Fourier transform libraries to take advantage of the high performance FFTs available on the Origin.
- Larry Takacs has determined the near optimal strip size for the physical parameterizations for the sequential and multitasked code. This strip size also seems to perform near optimally in the MP GEOS GCM.
- Romo, Snyder and Huang spent considerable effort in optimizing the single vector processor performance of the code on the NEC SX-4.

Even with these enhancements, the code still has modest absolute performance. More could still be done to the code, although the changes might require a considerable manpower investment.

5 Discussion

Since the MP GEOS GCM is compatible with GCM vc7.2 the benchmarks related here should give a good estimate of its true performance in later use. The only major deficiency of the code is that the GPIOS concept [12] has not yet been integrated, and thus all the benchmarks were run without history. Although initial GPIOS benchmarks [13, 14] seem promising, it is not yet clear whether the full-up MP GEOS GCM with history will perform acceptably in production.

On the whole the benchmarks indicate that the MP GEOS GCM parallelization was a success. On both the SGI Origin and the Cray T3E the simulation scales to large processor configurations. Figure 12 suggests that the MP GEOS GCM can achieve more than 30 days/day on 63 SGI O2K processors and more than 50 days/day on 256 Cray T3E processors for the high resolution ($360 \times$

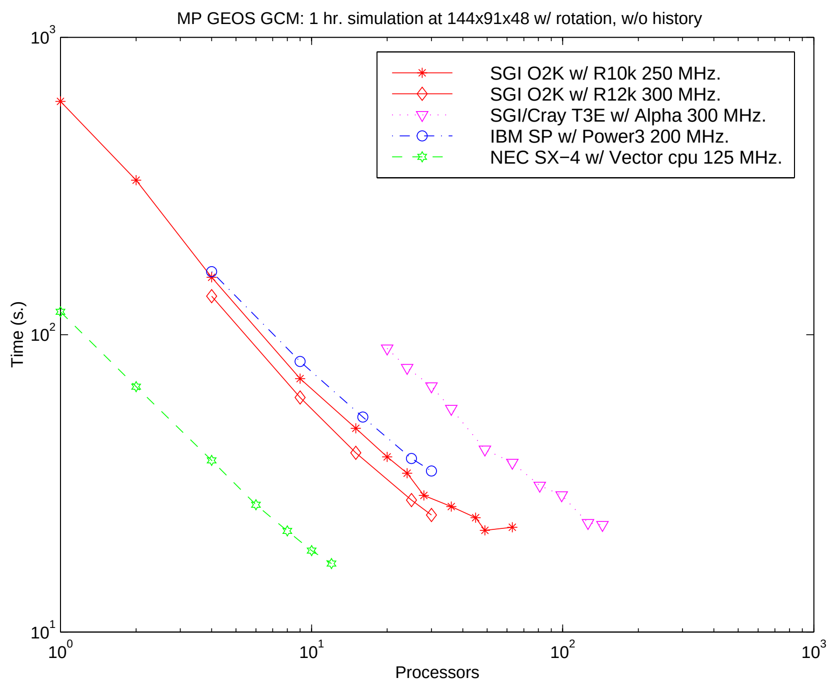

Figure 10:

181 × 48) case. This represents a large potential increase over the current multitasked GEOS GCM vc7.2 which runs on at most 32 SGI Origin 2000 processors.

The SGI Origin results are particularly encouraging. From Figure 11 it is apparent that the 1h simulation of $360 \times 181 \times 48$ resolution scales well to the full machine size (64 processors). Figure 10 indicates that the $144 \times 91 \times 48$ case saturates around 40 processors. This leads to the conclusion that the high resolution case might scale to 128 processors, particularly if the R10K 195 MHz processors were used.[6] From the limited tests on the R12K 300 MHz Origin, it appears that performance gains on that platform can be considerable for low numbers of processors.

The Cray T3E performs well on both problem sizes, scaling to about 256 processors for a $360 \times 181 \times 48$. The per processor performance is distinctly less than on the Origin, but the scalability is slightly better, perhaps due to the use of SHMEM communication in the rotation.[7] The lack of scalability above 256 could be a topic for future optimization if the T3E is seen as a viable platform for the code.

The NEC SX-4 achieves by far the best single processor performance. Although this might be expected from its customized vector processor, it was not a foregone conclusion: the SX-4 can only achieve such performances if the code vectorizes well. Not only was this the case with the Physics (which was largely unchanged from previous vector versions) but also, surprisingly, in the dynamics where the message-passing code restructuring was expected to inhibit large scale vectorization. The code has reasonable scalability to 12 processors. For a proper comparison, however, the shared memory parallel ("multitasked") GEOS GCM vc7.2 should be benchmarked on this machine as well. It is entirely possible that the production version, due to its heritage, would perform *as well if not better* than the MP GEOS GCM on the NEC SX-4. Clearly the NEC SX-4 would be a very desirable machine for production purposes if the DAO had access to one.

[6] Since the bottleneck in high processor runs is the interconnection network, higher processor performance brings no real benefit.

[7] A software problem on the SGI currently disallows the use of SHMEM in the rotation.

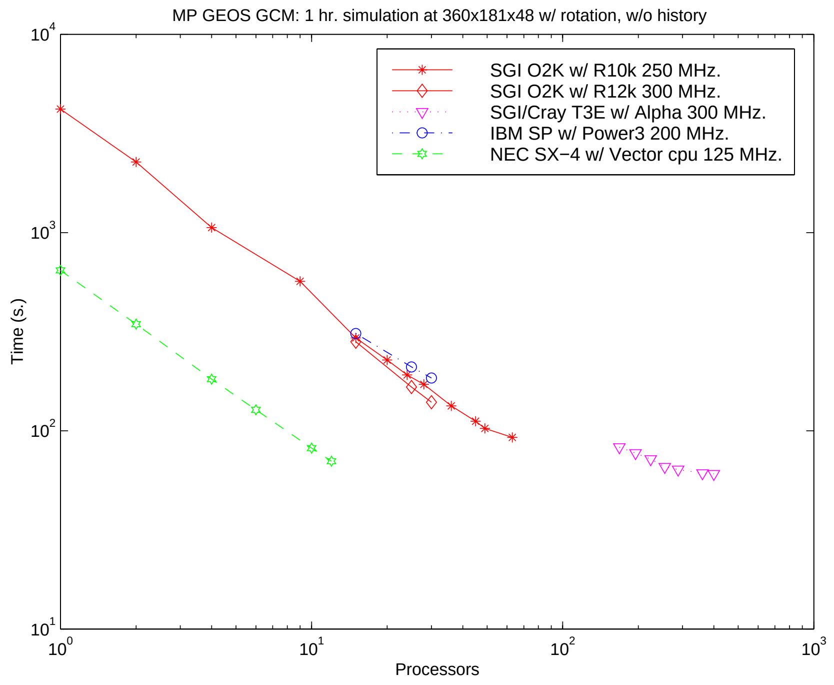

Figure 11: The times required for one hour of simulation with rotation but without stretching or history at high ($360 \times 181 \times 48$) resolution are given for all the benchmark platforms.

The IBM SP has in every case lower performance than the SGI Origin with R10K 250 MHz This is not necessarily a fair comparison, since compiler problems limited the degree of optimization used.[8] In addition, the MPI-only version was used, i.e., without the aid of low-latency Cray SHMEM primitives. IBM has its own proprietary low-latency LAPI primitives, which could conceivably be used for the critical communication as SHMEM has been used on the SGI Origin and the Cray T3E. Engineers at IBM have indicated that much optimization could still be done to improve performance on the IBM SP. This work should be considered pending and the benchmarks taken in proper perspective.

The absolute performance of the MP GEOS GCM does not come close to peak performance on any of the target platforms. The primary reason for this is not so much algorithmic as it is a software engineering issue. The GCM is a large and complicated code originally implemented on vector supercomputers. Experience shows that applications which are designed specifically for multiprocessors have the best chance for achieving a high percentage of peak performance. The physical parameterizations for long- and short-wave radiation, turbulence, moist processes and gravity wave drag are key examples. Although these are communication-free in MP GEOS GCM and their overall GFlop/s performance is higher than in the dynamical core, the ported, optimized versions still only attain a small fraction of peak performance. Expert opinions [17] tend to indicate that considerable optimization could still be done to the physics.

Near-peak performance can only be attained by extensive cache reuse. The stripping of vector-length arrays is a central concept to improving vector performance. Although a shorter array is likely to improve cache performance, the additional copy of a temporary array is a significant cost on cache-based machines. The removal of the stripping might actually increase overall performance, although this would be a radical, far-reaching change to the code.

Another reason for the performance deficiency is the lack of sufficient work for individual proces-

[8] Optimization level `-O2` had to be used instead of `-O4`

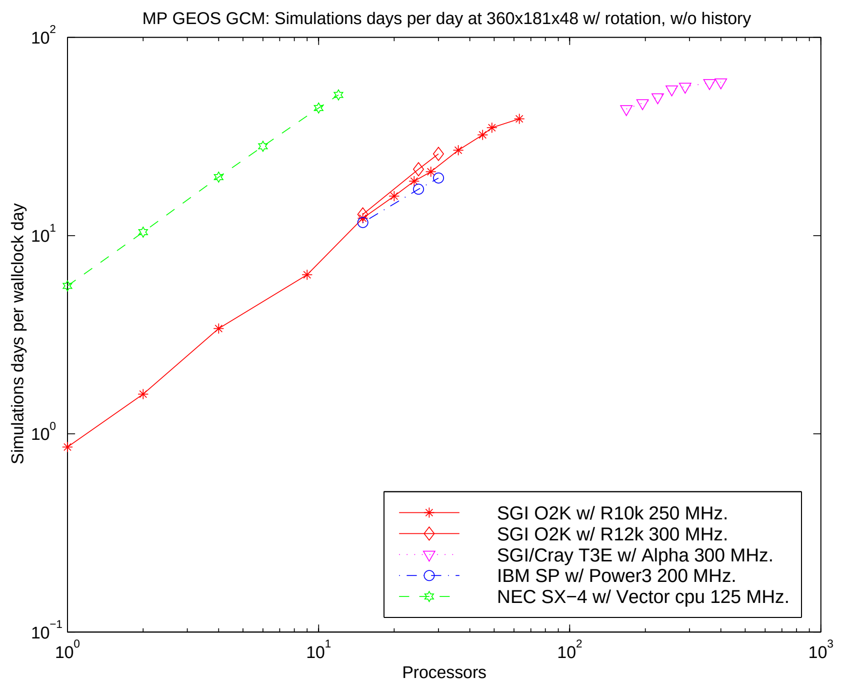

Figure 12: The number of model simulations days per wallclock day for the high ($360 \times 181 \times 48$) resolution is given for the benchmark platforms. Since other many issues arise in long simulations, they should be considered as estimates only.

sors for the problem sizes the DAO is interested in (currently maximum $360 \times 181 \times 48$). Good GCM performances have been obtained [18] using 32-bit real computation on resolutions as high as 1440x721x9. Unfortunately the DAO is obliged to calculate with 64-bit real precision and to use resolutions which have been tuned for scientifically valid output. Because the Courant-Friedrich-Levy condition requires that the dynamics core be called more often at higher resolutions, the dynamics already dominates the calculation at $360 \times 181 \times 48$ resolution. As indicated in the dynamics unit testing in Section 2 there is some redundant computation involved with maintaining boundary regions. Often they are calculated local to a processor from other data in order to avoid a communication. This means that the total number of operations performed in the code increases with the number of processors. This increase is substantial for high numbers of processors where the boundary regions make up a non-negligible portion of the local domain. Redundant calculation makes GFlop/s rates scale better than timings, making for example the dynamical core GFlop/s performance (Figure 5) seem quite acceptable, despite its scalability problems.

A related issue is the communication-to-computation ratio [19] for the column decomposition. In several GCM components, to wit the dynamical core and the C-to-A/A-to-C transformations, boundary regions have to be exchanged between processors. The complexity of the calculation is proportional to the number of points local to the processor, i.e., its "volume." The amount of boundary communication is related to the size of the boundary regions, that is, the column's "surface area." Conceptually — in terms of the problem's *iso-efficiency* [20] — each of the horizontal dimensions needs to be increased by a factor $\sqrt{p}$, here p is the number of processors, to keep the ratio between work and communication constant.[9] Clearly given the DAO's obligation to hold resolution at $360 \times 181 \times 48$ for the time being, iso-efficiency cannot be maintained for increasing numbers of processors.

The communication-to-computation issue is not as imposing as it sounds. The optimizations to

[9] In other words, the number of points per processor needs to be kept constant.

the A-to-C and C-to-A transformations perform the calculation to the local column "core" while the boundary regions are en-route. As long as there is sufficient work in the local calculation, the communication can be hidden or "overlapped" and performance is highly scalable (see Figure 7). Conceptually the technique could also be applied to parts of the dynamical core. However, the complexity of this approach — the points have to be split into sets which are and are not affected by a remote processor's data — would require a complete re-write of the dynamical core. Given the acceptable performance and scalability of the dynamical core along with current developments [21] to parallelize a replacement, there is little motivation to take this step.

Finally, the MP GEOS GCM is a highly synchronous code — synchronization is required at numerous locations in the dynamics, for example every time boundary regions are exchanged, or a rotation is performed. These synchronization points are not only barriers, but also blocking receives, send_receives, waiting for the completion of non-blocking communication, and collective communication. On large processor configurations minor load imbalances or "process skewing" can cause many processors to wait at the synchronization point. In addition, global synchronization primitives are not necessarily optimally implemented on the SGI Origin [22] for example. Execution tracing indicates that the synchronization overhead makes up a large part of the overall execution time for high processor runs, indicating that one or several of these issues is arising. The synchronization issues are very much related to the target platform, its interconnection network and number of processors. Once the target platform is identified it would be worthwhile to address these issues to enhance code performance.

Acknowledgments

We are indebted to John Snyder, John Romo, Denise Huang and Nick Chepurniy for their performance enhancements on the NEC SX-4, to Samson Cheung for his benchmarking and analysis on the SGI Origin 2000, to Eric Myra for his help in porting the code to the IBM, and to Tom Clune and Spencer Swift for their outstanding support of development on the Cray T3E. We thank SiliconGraphics, Internation Business Machines, and Environment Canada for giving us access to their machines for benchmarking. The work of William Sawyer was partially funded by the NASA High Performance Computing and Communications Earth and Space Sciences (HPCC ESS) project.

References

[1] L. L. Takacs, A. Molod, and T. Wang. Documentation of the Goddard Earth Observing System (GEOS) General Circulation Model — Version 1. Technical Memorandum 104606, NASA, Code 910.3 Greenbelt MD, 20771, USA, 1994.

[2] W. Sawyer, R. Lucchesi, P. Lyster, L.L. Takacs, J. Larson, A. Molod, S. Nebuda, and C. Pabon-Ortiz. Parallelization of the DAO Atmospheric General Circulation Model. In B. Kågstrøm, J. Dongarra, E. Elmroth, and J. Wasniewski, editors, *Applied Parallel Computing, 4th International Workshop, PARA'98*, pages 510–514. Springer-Verlag, 1998. Lecture Notes in Computer Science 1541.

[3] W. Sawyer, R. Lucchesi, L.L. Takacs, and A. Molod. Modular Fortran 90 Implementation of a Parallel Atmospheric General Circulation Model. In unknown, editor, *Proceedings of EUROPAR'99*. Springer-Verlag, 1999. Lecture Notes in Computer Science.

[4] M. Fox-Rabinovitz, L. L. Takacs, G. Stenchikov, M. Suárez, and R. C. Govindaraju. A Variable Resolution GCM Dynamical Core with the Real Orography. *Mon. Wea. Rev.*, 1999. Accepted for publication.

[5] Peter M. Lyster, William Sawyer, and Lawrence L. Takacs. Design of the Goddard Earth Observing System (GEOS) Parallel General Circulation Model (GCM) . DAO Office Note 97-13, Data Assimilation Office, NASA, Code 910.3 Greenbelt MD, 20771, USA, 1997.

[6] W. Sawyer, L. L. Takacs, A. Molod, and R. Lucchesi. Data and Procedural Design of the Message-Passing GEOS General Circulation Model. DAO Office Note 1999-xxx, Data Assimilation Office, NASA, Code 910.3 Greenbelt MD, 20771, USA, 1999. Not yet approved by the Configuration Control Board.

[7] W. Sawyer. Reproducibility Issues: Sequential vs. MPI GCM. `http://dao/Intranet/GEOS3/Software/Core/GCM/Design/reprod.html`, 1997.

[8] W. Sawyer, P. M. Lyster, A. da Silva, and L. L. Takacs. Parallel Grid Manipulations in Earth Science Calculations. In José M. Laginha M. Palma, editor, *3rd International Meeting on Vector and Parallel Processing*. Springer-Verlag, 1998. Lecture Notes in Computer Science.

[9] W. Sawyer. MPI GEOS DAS Core Prototyping. `http://dao.gsfc.nasa.gov/Intranet/GEOS3/Software/Core/Walkthroughs/98%0510-AdvPanel/`, May 1998.

[10] M. J. Suárez and L. L. Takacs. Documentation of the ARIES/GEOS Dynamical Core: Version 2. NASA Technical Memorandum 104606, NASA, Code 910.3 Greenbelt MD, 20771, USA, 1995.

[11] L. L. Takacs, W. Sawyer, M. J. Suarez, and M. S. Fox-Rabinovitz. Filtering Techniques on a Stretched Grid General Circulation Model. NASA Technical Memorandum 104606, NASA Code 910.3, 1999. In preparation.

[12] R. Lucchesi. Requirements and Design of the GEOS-3 Parallel I/O Subsystem. `http://dao/Intranet/GEOS3/Software/Core/GPIOS/parIO/`, 1997.

[13] R. Lucchesi. I/O Parallelization for the Goddard Earth Observing System Data Assimilation System (GEOS DAS). `http://dao.gsfc.nasa.gov/DAO_people/lucchesi/Arizona98/`, June 1998.

[14] R. Lucchesi. I/O Parallelization for the Goddard Earth Observing System Data Assimilation System (GEOS DAS). `http://dao.gsfc.nasa.gov/DAO_people/lucchesi/CAS98/`, August 1998.

[15] L. L. Takacs, M. Fox-Rabinovitz, M. J. Suarez, and W. Sawyer. Filtering Techniques in the Stretched-grid GEOS-2 GCM. In *Proceedings of 13th Conference on Numerical Weather Prediction*. AMS, 1999.

[16] L. L. Takacs. Private communication. Comparison of leap-frog and Matsuno schemes, 1999.

[17] J. A. Abeles. Private communication. Discussion on GEOS GCM performance, 1997.

[18] D. S. Schaffer and M. J. Suárez. Next Stop: Teraflop; The Parallelization of an Atmospheric General Circulation Model. In *High Performance Computing Symposium 98 Proceedings*, April 1998.

[19] Alan Chalmers and Jonathan Tidmus. *Practical Parallel Processing: An introduction to problem solving in parallel.* International Thomson Computer Press, London, March 1996.

[20] A. Y. Grama, A. Gupta, and V. Kumar. Isoefficiency: measuring the scalability of parallel algorithms and architectures. *IEEE parallel and distributed technology: systems and applications*, 1(3):12–21, August 1993.

[21] S.-J. Lin and R. B. Rood. A Flux-Form Semi-Lagrangian General Circulation Model with a Lagrangian Control-Volume Vertical Coordinate. In *Proceedings of the Rossby-100 Symposium, Stockholm, Sweden*, December 1999. Awaiting publication.

[22] B. Nelson. Private communication. E-mail discussion, 1999.